Remarks

The drawings and specification have been amended to illustrate and recite that management server 140 includes a CPU 231, operating system 232, RAM 233 and storage 234, and server 145 includes a CPU 331, operating system 332, RAM 333 and storage 334. This is inherent in and well known from the term "server" used throughout the drawings and specification. Also, the specification recites "The servers 140-155 may implement one or more server technologies including, for example UNIX, ...Novel ... Windows NT." These are examples of the operating systems 232 and 332. Also, it is well known that to install and execute an operating system requires a CPU, RAM and storage. Also, the drawings and specification illustrate and describe various modules which include program functions that require a CPU, RAM and storage.

Claims 22-37 have been canceled and new claims 38-49 presented. In a telephonic interview with Examiner Shaq Taha on August 18, 2008, the Examiner indicated that the prior rejections under 35 USC 102 and 35 USC 103 based on US Patent 6,801,949 to Bruck and US Publication 2002/0032768 by Voskuil have been overcome and withdrawn. In the Office Action of August 1, 2008, claims 22-37 were rejected under 35 USC 103(a) based on combinations of US Patent 7,080,378 to Noland et al. with US Publication 2003/0014507 by Bertram et al., US Publication 2003/0051187 by Mashayekhi et al., US Patent 7,177,901 to Dutta et al. and US 6,859,798 to Bedell et al. Applicants respectfully traverses these current rejections as applied to new claims 38-49, based on the following.

Claim 38 recites a method for allocating a real server to a pool of real servers. Performance data of a first real server for execution of an application is automatically sent by the first real server to a second real server. The first server is part of the pool of servers. Based on the performance data, the second server automatically determines that the first server is functional but has reached a predetermined upper level of utilization. In response, there is an automatic identification of another, available real server having the application but not currently allocated to the pool. There is an automatic request via a network from the other real server for connection settings for the other real server. There is an automatic reconfiguration of the second

server to identify the other real server as part of the pool and record the connection settings for communication with the other real server. There is automatic sending, via a network, of a reconfiguration request to a load balancer for the pool to allocate the other real server to the pool for handling requests for the application.

Noland et al. disclose:

"If the deployed **virtual servers** are close to near-term service saturation, the software will create and deploy a **new virtual server** with identical applications (step 504), and then direct the service request to this new server (step 505). This new virtual server can be deployed and activated via the method depicted in FIG. 3. More than one new virtual server may be deployed and activated based on the saturation condition that is detected (e.g., rate of saturation, number of servers approaching saturation simultaneously, etc.)." (Emphasis added) Noland et al. Column 5 lines 24-36.

"Referring to FIG. 3, a flowchart illustrating the process of deploying a virtual server by means of an automated software solution is depicted in accordance with the present invention. The process begins when the user logs into the software (step 301) and selects definition criteria for the newly deployed system image (step 302). The definition criteria include the following: a pool of TCP/IP addresses that the software assigns to newly deployed virtual servers, a pool of names the software assigns to the new virtual servers, and a model image that is used as a target image for the creation and deployment of the new virtual server. The model image is the current contents of memory, including the operating system and running programs. Every new virtual server is an exact copy of the model image, except for the dynamic network and server definitions that identify the new server as a unique entity. The user verifies the definition criteria and clicks a "submit" link (step 303). This link contained an imbedded common sequence that requests the software to rapidly deploy a new virtual server. Alternatively, the step of deploying the new virtual server may be automated and triggered by specified event, e.g. reaching a saturation threshold on currently deployed servers (as described in more detail below).

Furthermore, **the new virtual server can actually be a subset of the original server that focuses on a critical portion of the process**. This can be done in several ways. For example, **the virtual server can be composed of a series of linked server processes that are individually utilized in the overall server process. When one of the sub processes becomes a bottleneck, that sub process could be cloned as necessary to eliminate the bottleneck.** In response to the request from the user, **the software automatically updates the VM system directory to include the new virtual server** (step 304), prepares the virtual server media (disk allocations) (step 305), propagates the server model image into the new virtual server (step 306), updates the new image with local identification parameters (step 307), and then boots the new server (step 308). After the new virtual server is deployed, the end user simply clicks another link in order to interface with the new server (step 309). Alternatively, the new server may be automatically integrated into a preexisting server cluster. The entire process of deploying a new virtual server by means of the present invention can be fully automated. When done manually, it takes less than five minutes." (Emphasis added) Noland et al. to Column 3l line 56 to Column 4 line 33.

"Referring now to FIG. 4, a schematic diagram illustrating the architecture of the virtual machine environment of the automated software solution is depicted in accordance with the present invention. The present example is described within the context of the SnapVantage software solution, but it should be pointed out that the features of the present invention may be implemented by means of other software solutions. **SnapVantage VM server 401 is a Virtual service machine that manages the cloning process of Linux images**, i.e. Model Images 405 and 406. **This cloning process uses a Shared Virtual Array Administrator (SVAA) 407 in order to create array of cloned virtual servers 408**. SnapVantage runs disconnected and communicates to clients 409 and 410 via TCP/IP 404. The SnapVantage web server 402 is the location of the web pages used by the SnapVantage GUI on client 409, and executes under a local Apache (or other) web server. The local deployment application 403 is the user created code imbedded in local web pages that drives specific SnapVantage functions. This

component is deployed in environments that choose to allow end users to define a new virtual server. Referring now to FIG. 5, a flowchart illustrating the process of load balancing among virtual servers is depicted in accordance with the present invention. The present invention provides a software-based solution that utilizes a communication mechanism in either direction and monitors the load status of deployed servers." (Emphasis added) Noland et al. Column 4 line 36 to Column 5 line 4.

Thus, Noland et al. pertain to a virtual machine environment, where virtual servers are formed on the same real computer by cloning. In contrast to independent claim 38, Noland et al do not teach or suggest automatic addition of another, **real server** to a server pool. To clone a virtual machine is a much different concept and implementation than allocating another real server to a server pool because the other real server is external, and the allocation of the other real server to the server pool involves external investigations and reconfigurations, both to the second server, the other real server to be added to the server pool and the load balancer. Because Noland et al. is confined to a virtual machine environment in a single real server, Noland et al. do not teach or suggest the allocation of another real server to an existing pool or real servers, or the remaining steps of claim 38, either: In response to the performance data indicating that the first server is functional but has reached a predetermined upper level of utilization, there is an automatic identification of another, available real server having the application but not currently allocated to the pool. There is an automatic request via a network from the other real server for connection settings for the other real server. There is an automatic reconfiguration of the second server to identify the other real server as part of the pool and record the connection settings for communication with the other real server. There is automatic sending, via a network, of a reconfiguration request to a load balancer for the pool to allocate the other real server to the pool for handling requests for the application.

Bertram et al. disclose,

"Performance data are analyzed for each of the nodes in the cluster, via step 224. Step 224 thus analyzed data for the nodes 112 and 114 or the nodes 132, 134 and 136 of the

cluster 110 and 130, respectively. Step 224 includes diagnosing bottlenecks for each node." Bertram et al. Paragraph [0029]

"It is then determined if the selected computer system, or node, is part of a cluster, via step 258. If so, it is determined whether there are more nodes in the cluster, via step 259. If so, then the next node is selected, via step 260. Steps 256 through 260 are then repeated until the performance data for each of the nodes in the cluster has been analyzed." Bertram et al. Paragraph [0035]

"It is then determined whether a bottleneck object has been created for the selected computer system, via step 262. If so and the selected computer system is part of a cluster, then information about the other, companion nodes in a cluster is added to the bottleneck object, via step 262. The information added in step 264 includes setting four counters for each companion node in the cluster. If the current node is down, then the first counter for each companion node is set to a one. If the current node is bottlenecked, then the second counter for each companion node is set to a one. If the companion node can absorb all of the workload from the (current) bottlenecked node, then the third counter is set to a one. If the companion node can absorb all of the workload from the (current) bottlenecked node only with other nodes, then the fourth counter for the companion node is set to a one. If not set to a one, then the counter remains a zero. Thus, information relating to companion nodes in the cluster is accounted for in the bottlenecked object for the current node. In addition, when the companion nodes in the cluster are later analyzed, information for the current node is accounted for. Thus, nodes in a cluster are analyzed from two perspectives-from the nodes own perspective and from the perspective of other nodes in the cluster." Bertram et al. Paragraph [0036]

Thus, all of the nodes of Bertram et al. that are analyzed are already part of a cluster, and Bertram et al. do not disclose addition of a server to the cluster or how that is accomplished. Therefore, Bertram et al. do not pertain to the present invention as recited in claim 38, and do not fill the gap of the other cited prior art.

Mashayekhi et al. disclose a method of assigning a failover node for a failed node.

"Most operating systems have performance monitors or indictors that track the usage of system entities, such as memory, disk network, and processor, amongst others." Mashayekhi et al. Paragraph [0037]

"Using the APIs, an application can observe and track its use of system entities and perform operations based on the value of the performance counters during its operation. These performance indicators can be accessed and used to assess possible failover nodes. For example, in a cluster, should the memory usage of applications running on a node be above 75% of that node's physical memory, it can be designated as not a suitable failover site for a failed node. A node that is showing only 25% memory usage, however, can be designated as a suitable failover site." Mashayekhi et al. Paragraph [0037].

"After failure of node n is detected, at step 251 the weight of nodes n+1 and n-1 are determined using one or more predetermined performance indicators and a predefined equation as described above. Once the weight of each node is ascertained, a determination is then made at step 252 as to whether node n+1 or n-1 has the lowest weight. If node n+1 has the lowest weight, that node is established as the failover node (step 253), but if node n-1 has the lowest weight, node n-1 is established as the failover node (step 254). Thus, a more intelligent failover node can be chosen, thereby improving the high availability of the cluster system." Mashayekhi et al. Paragraph [0039]

"Yet another failover policy can be implemented in a cluster environment in which any single node may failover to any surviving node. Any type of hash function that randomly selects a failover node from the available nodes can be used in this type of system." Mashayekhi et al. Paragraph [0041]

"Other failover polices ... include first-in-first-out (FIFO) or Last-in-First-out (LIFO) policies, wherein a failover node is designated according to the order in which nodes joined the cluster." Mashayekhi et al. Paragraph [0042]

"Yet another failover policy according to the present disclosure involves maintaining a prioritized queue of failover nodes." Mashayekhi et al. Paragraph [0043]

Thus, Mashayekhi et al. are concerned with identifying a failover node, and not how to add a server to a cluster that has reached an upper level of utilization. Moreover, Mashayekhi et al. are concerned with failed nodes, and not a functional node, as recited in claim 38, that has reached an upper level of utilization. Therefore, Mashayekhi et al. do not teach or suggest the present invention as recited in claim 38 and do not fill the gap of the other cited prior art.

Dutta discloses a method, system and computer program product to redirect requests from content servers to load distribution servers.

"By placing identical copies on multiple servers, a re-directing arrangement of load servers and content servers may be implemented which will allow a load distribution server to protect any given content server from being overloaded with file requests." Dutta Column 4 lines 44-48.

"In a system where a central load distribution server at a publicized URL redirects requests for files to a number of content servers holding identical content on the basis of dynamically determined capacity utilization of those servers, clients are prevented from directly accessing one of the content servers without first being redirected from the central load distribution server. In the event that a client attempts to access one of the content servers without first having been redirected there from the load distribution server, the client is redirected to a page containing a notice of the error, then redirected yet again to the load distribution server." Dutta Column 3 lines 21-32.

Thus, Dutta is concerned with redirecting requests from a central load distribution server to a pre-existing cluster of content servers to prevent direct access by users to the content servers, and not allocating a new server to the cluster. Dutta does not teach how to add a new server to the cluster. Therefore, Dutta does not teach or suggest the present invention and does not fill the gap of the other cited prior art.

Bedell et al. disclose

"a reporting system 100 by which a variety of data resources may be accessed for business analytic, report generation and other intelligence purposes." Bedell et al. Column 5 lines 10-13.

"Through using the reporting system 100 ..., analysts, managers and other users may query or interrogate a plurality of databases or database arrays to extract demographic, sales, and/or financial data and information and other patterns from records stored in such databases or database arrays to identify strategic trends. Those strategic trends may not be discernible without processing the queries and treating the results of the data extraction according to the techniques performed by the systems and methods of the invention. This is in part because the size and complexity of some data portfolios stored in such database or database arrays may mask those trends." Bedell et al. Column 5 lines 22-33.

Thus, Bedell et al. are concerned with report generation for demographic, sales and/or financial data, and therefore are unrelated to the present invention. Bedell et al. should not even be combined with the other cited prior art, because the subject matter is so different.

In summary, none of the cited prior art teaches the automatic addition of a real server to a pool of real servers when an existing real server in the pool is functional but reaches a predetermined upper level of utilization. Also, none of the cited prior art teaches the technique of claim 38 for adding the real server to the server pool.

The other independent claims 42 and 46 distinguish over the cited prior art for the same reasons that claim 38 distinguishes thereover.

Therefore, no rejection under 35 USC 103(a) should be made for new claims 38-49.

Claim 39 depends on claim 38 and further distinguishes over the cited prior art by the step of automatically requesting, via a network from the other real server, port authentication settings for the other real server; and wherein the step of automatically reconfiguring the second server includes the step of recording the authentication settings. This is not taught or suggested by the prior art. Claims 43 and 47 similarly, further distinguish over the cited prior art.

Claim 40 further distinguishes over the prior art by the subsequent steps of based on subsequent performance data, determining by the second (real) server that the first (real) server is functional but under utilized such that the first server is no longer needed in the pool, and in response, automatically de-allocating the first server from the pool (or real servers). Claims 44 and 48 similarly, further distinguish over the cited prior art.

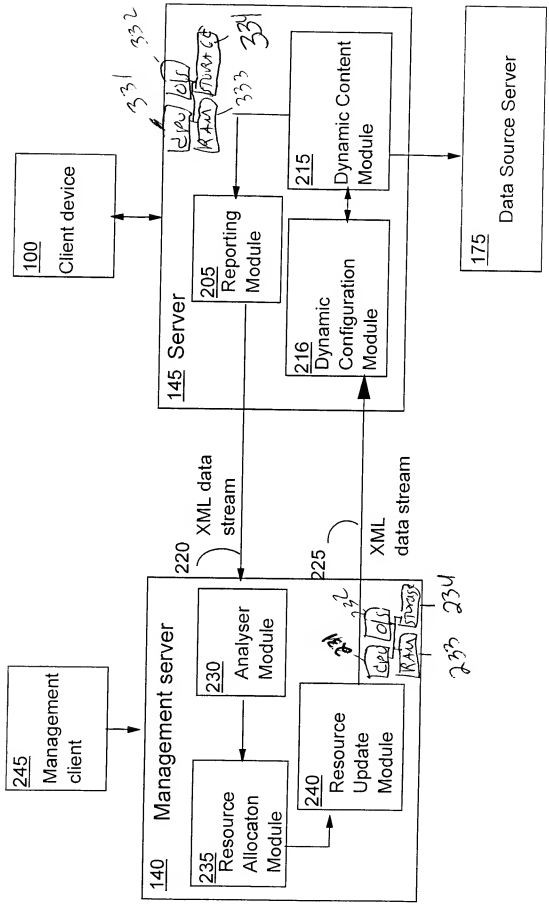
Based on the foregoing, Applicants request allowance of the present patent application as amended above.

Respectfully submitted,

Dated: 08/28/2008
Telephone: 607-429-4368
Fax No.: 607-429-4119

/Arthur J. Samodovitz/
Arthur J. Samodovitz
Reg. No. 31,297

Fig 2


245
Management client
140
Management server
235
Resource Allocaton Module
230
Analyser Module
240
Resource Update Module
220
XML data stream
225
XML data stream
100
Client device
145
Server
205
Reporting Module
216
Dynamic Configuration Module
215
Dynamic Content Module
175
Data Source Server